# Laboratorul 04.

## Problema 1

Se ofera clasa de baza ***Patrulater*** care descrie o figura geometrica reprezentata printr-un patrulater convex. Pornind de la clasa oferita, implementati urmatoarea ierarhie de clase, alegand constructorii considerati potriviti pentru fiecare clasa:

- clasa ***Paralelogram*** care mosteneste ***Patrulater***;
- clasele ***Romb*** si ***Dreptunghi*** care mostenesc clasa ***Paralelogram***;
- clasa ***Patrat*** care mosteneste clasa ***Dreptunghi***.

Implementati in fiecare clasa, exceptand clasa ***Patrulater***, o metoda care calculeaza aria figurii geometrice, uzitand formula specifica. De asemenea, creati o clasa executabila pentru testarea claselor implementate.

Clasa Romb contine ca date, pe langa cele din clasa parinte, si dimensiunile diagonalelor.

```
Arie paralelogram = lat1 * lat2 * sin(unghi1)
Arie romb = (diag1 * diag2) / 2
Arie dreptunghi = lungime * latime
Arie patrat = latura * latura
```

```
class Patrulater {
    public int latura1, latura2, latura3, latura4;
    public double unghi1, unghi2, unghi3, unghi4;

    public Patrulater() {
        this(0, 0, 0, 0);
    }

    public Patrulater(int latura1, int latura2, int latura3, int latura4) {
        this.latura1 = latura1;
        this.latura2 = latura2;
        this.latura3 = latura3;
        this.latura4 = latura4;
    }

    public Patrulater(double unghi1, double unghi2, double unghi3, double unghi4) {
        this(0, 0, 0, 0, unghi1, unghi2, unghi3, unghi4);
    }

    public Patrulater(int latura1, int latura2, int latura3, int latura4,
                      double unghi1, double unghi2, double unghi3, double unghi4) {
        this(latura1, latura2, latura3, latura4);
        this.unghi1 = unghi1;
        this.unghi2 = unghi2;
        this.unghi3 = unghi3;
        this.unghi4 = unghi4;
    }

    public int perimetru() {
        int result;
        result = latura1 + latura2 + latura3 + latura4;
        return result;
    }
}
```

## Problema 2

Pornind de la clasa de baza ***Array*** oferita, implementati urmatoarele clase:

- clasa ***SortedArray*** care modeleaza un vector de numere intregi sortat crescator, folosind mostenirea;
- clasa ***MyStack*** care modeleaza o stiva care contine numere intregi, folosind agregarea.

Ambele clase vor utiliza metodele puse la dispozitie in clasa ***Array***, in forma originala sau intr-o forma modificata, iar clasa ***MyStack*** trebuie sa ofere metodele ***push*** si ***pop***, specifice acestei structuri de date. Metoda ***push*** va oferi posibilitatea introducerii unui numar intreg in varful stivei, in timp ce metoda ***pop*** va inlatura elementul din varful stivei si il va intoarce.

```
public class Array {
    //Vectorul in care se vor retine elementele
    private Vector vector;

    //Constructor clasei
    public Array() {
        //Instantierea vectorului cu elemente
        vector = new Vector();
    }

    //Metoda care adauga un element in vector, folosind pozitia curenta
    public void addElement(Integer x) {
        vector.add(x);
    }

    //Metoda care adauga un element in vector, tinand cont de pozitia indicata
    public void addElement(Integer x, int poz) {
        if(poz >= 0 && poz <= vector.size()) {
            vector.add(poz, x);
        }
    }

    //Metoda care returneaza elementul aflat in vector la pozitia indicata
    public int get(int poz) {
        int result;
        if(poz >= 0 && poz < vector.size()) {
            result = (int) vector.get(poz);
            return result;
        } else {
            return Integer.MIN_VALUE;
        }
    }

    //Metoda ce intoarce numarul de elemente din vector
    public int getSize() {
        return vector.size();
    }

    //Metoda pentru stergerea unui element din vector
    public boolean remove(Integer x) {
        return vector.remove(x);
    }

    //Metoda pentru stergerea elementului de pe pozitia pos din vector
    public Integer remove(int pos) {
        return (Integer) vector.remove(pos);
    }

    //Metoda uzitata pentru afisarea unui obiect de tip Array
    public String toString() {
        String result = "{";
        for(int i = 0; i < vector.size(); i++) {
            result += get(i) + ", ";
        }
        result += "}";
        return result;
    }

    public void sort() {
        Collections.sort(vector);
    }
}
```

Creati o clasa ce contine un main pentru testarea claselor implementate.

```
Collections.sort
```

## Problema 3

Sa se implementeze o clasa ***HSet*** care modeleaza o multime realizata ca tabel de dispersie. Clasa este derivata din Hashtable [http://docs.oracle.com/javase/7/docs/api/java/util/Hashtable.html] si contine metodele: ***add***, ***remove***, ***toString***.

Cheia si valoarea vor fi egale (cheile sunt elementele multimii). Pentru testare, folositi clasa ***Test3***.

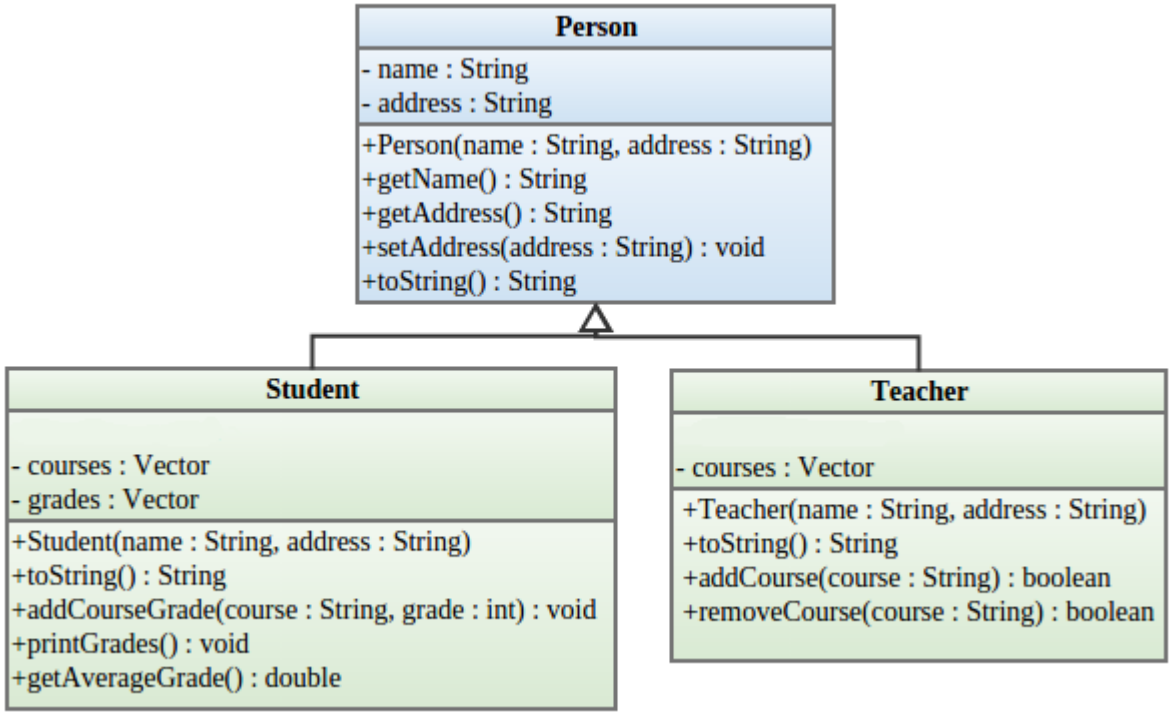
```
//Adauga un element in multime, daca nu exista deja
public boolean add(Object value);
//returneaza un String cu elementele multimii (doar cheile, nu perechi)
public String toString();
//Sterge perechea corespunzatoare cheii, intorcand valoarea
public Object remove(Object key);
```

Se vor supradefini doar metodele care necesita acest lucru!

```
class Test3 {
    public static void main(String args[]) {
        HSet set = new HSet();
        set.add("Laborator");
        set.add("Agregare");
        set.add("Mostenire");
        System.out.println(set);
        System.out.println(set.size());
        set.add("Laborator");
        if(set.size() == 4) {
            System.out.println("Multimea nu trebuie sa contina duplicate!");
        }
        System.out.println(set.remove("POO"));
        System.out.println(set.remove("Laborator"));
        if(set.size() != 2) {
            System.out.println("Stergerea nu functioneaza!");
        }
        set.add("Supradefinire");
        set.add("Supraincarcare");
        System.out.println(set);
    }
}
```

## Problema 4

Sa se implementeze ierarhia de clase descrisa in figura de mai jos.




Pentru testare, se poate folosi clasa ***Test04***. Metoda ***addCourse*** din clasa ***Teacher*** verifica daca a fost deja asignat cursul respectiv profesorului si intoarce ***true*** doar daca acesta nu exista si a fost adaugat. De asemenea, metoda ***removeCourse*** intoarce ***true*** doar daca profesorul avea asignat cursul respectiv si s-a putut realiza stergerea.

Atentie la modul in care va definiti constructorii claselor copil (***Student*** si ***Teacher***)!

```
super
```

```
class Test4 {
    public static void main(String args[]) {
        Person student, teacher, person;
        student = new Student("Popescu Ion", "Bucuresti");
        teacher = new Teacher("Ionescu Gigel", "Bucuresti");
        person = new Person("Maria", "Iasi");
        assert (person.getName().equals("Maria")) : "Metoda getName din clasa Person nu este implementata corect";
        assert (((Teacher) teacher).addCourse("Programare")) : "Metoda addCourse din clasa Teacher nu este " +
                "implementata corect";
        assert (((Teacher) teacher).addCourse("Algoritmica")) : "Metoda addCourse din clasa Teacher nu este " +
                "implementata corect";
        assert (((Teacher) teacher).addCourse("Matematica")) : "Metoda addCourse din clasa Teacher nu este " +
                "implementata corect";
        assert (!((Teacher) teacher).addCourse("Programare")) : "Metoda addCourse din clasa Teacher nu este " +
                "implementata corect";
        assert (((Teacher) teacher).removeCourse("Programare")) : "Metoda addCourse din clasa Teacher nu este " +
                "implementata corect";
        assert (!((Teacher) teacher).addCourse("Programare")) : "Metoda addCourse din clasa Teacher nu este " +
                "implementata corect";
        ((Student) student).addCourseGrade("Programare", 10);
        ((Student) student).addCourseGrade("Algoritmica", 9);
        ((Student) student).addCourseGrade("Matematica", 8);
        assert (Math.abs(((Student) student).getAverageGrade() - 9.00) <= 0.001) : "Metoda getAverageGrade din clasa " +
                "Student nu a fost implementat corect";
        ((Student) student).printGrades();
        //Ce metoda toString se va apela? Din ce clasa?
        System.out.println(student);
        System.out.println(person);
        System.out.println("Felicitari! Problema a fost rezolvata corect!");
    }
}
```